The command line is composed of:

- Name of the requested verb.
- Transaction identifier, correlates commands and responses. Transaction identifiers may have values between 1 and 999999999 and transaction identifiers are not reused sooner than 3 minutes after completion of the previous command in which the identifier was used.
- Name of the endpoint that should execute the command (in notifications, the name of the endpoint that is issuing the notification).
- Protocol version.

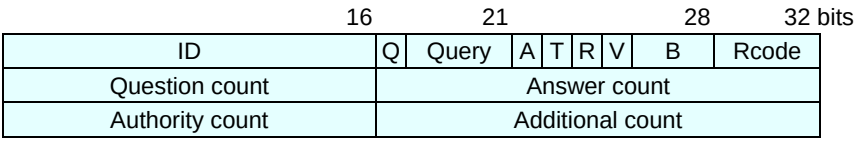
These four items are encoded as strings of printable ASCII characters, separated by white spaces, i.e. the ASCII space (0x20) or tabulation (0x09) characters. It is recommended to use exactly one ASCII space separator.

DNS

RFC 1035 1987-11 http://www.cis.ohio-state.edu/htbin/rfc/rfc1035.html
RFC 1706 1994-01 http://www.cis.ohio-state.edu/htbin/rfc/rfc1706.html

The Domain Name Service (DNS) protocol searches for resources using a database distributed among different name servers.

The DNS message header structure is shown in the following illustration:



DNS message header structure

ID
16-bit field used to correlate queries and responses.

Q
1-bit field that identifies the message as a query or response.

Query
4-bit field that describes the type of message:
0 Standard query (name to address).
1 Inverse query (address to name).
2 Server status request.

A
Authoritative Answer. 1-bit field. When set to 1, identifies the response as one made by an authoritative name server.

T
Truncation. 1-bit field. When set to 1, indicates the message has been truncated.

R
1-bit field. Set to 1 by the resolve to request recursive service by the name server.

V
1-bit field. Signals the availability of recursive service by the name server.

B
3-bit field. Reserved for future use. Must be set to 0.

RCode
Response Code. 4-bit field that is set by the name server to identify the status of the query:

0 No error condition.
1 Unable to interpret query due to format error.
2 Unable to process due to server failure.
3 Name in query does not exist.
4 Type of query not supported.
5 Query refused.

Question count
16-bit field that defines the number of entries in the question section.

Answer count
16-bit field that defines the number of resource records in the answer section.

Authority count
16-bit field that defines the number of name server resource records in the authority section.

Additional count
16-bit field that defines the number of resource records in the additional records section.

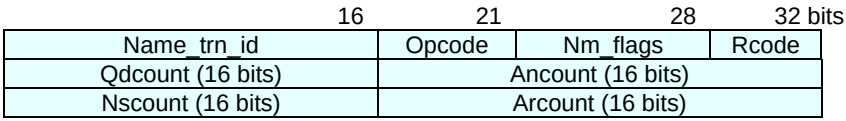
NetBIOS/IP

IETF RFC 1002 http://www.cis.ohio-state.edu/htbin/rfc/rfc1002.html

NetBIOS/IP is a standard protocol to support NetBIOS services in a TCP/IP environment. Both local network and Internet operations are supported. Various node types are defined to accommodate local and Internet topologies and to allow operation with or without the use of IP broadcast.

NetBIOS types may be Name Service, Session or Datagram.

The format of the header is shown in the following illustration:



NetBIOS/IP header structure

Name_trn_id
Transaction ID for the Name Service Transaction.

Opcode
Packet type code: Possible values are:

0 Query.
5 Registration.
6 Release.
7 WACK.
8 Refresh.

Nm_flags
Flags for operation.

Rcode
Result codes of request.

Qdcount
Unsigned 16 bit integer specifying the number of entries in the question section of a name.

Ancount
Unsigned 16 bit integer specifying the number of resource records in the answer section of a name service packet.

Nscount
Unsigned 16 bit integer specifying the number of resource records in the authority section of a name service packet.

Arcount
Unsigned 16 bit integer specifying the number of resource records in the additional records section of a name service packet.

FTP

IETF RFC 959 1985-10 http://www.cis.ohio-state.edu/htbin/rfc/rfc959.html

The File Transfer Protocol (FTP) provides the basic elements of file sharing between hosts. FTP uses TCP to create a virtual connection for control information and then creates a separate TCP connection for data transfers. The control connection uses an image of the TELNET protocol to exchange commands and messages between hosts.

Commands

FTP control frames are TELNET exchanges and can contain TELNET commands and option negotiation. However, most FTP control frames are simple ASCII text and can be classified as FTP commands or FTP messages. The standard FTP commands are as follows:

Command	*Description*
ABOR	Abort data connection process.
ACCT <account>	Account for system privileges.
ALLO <bytes>	Allocate bytes for file storage on server.
APPE <filename>	Append file to file of same name on server.
CDUP <dir path>	Change to parent directory on server.
CWD <dir path>	Change working directory on server.
DELE <filename>	Delete specified file on server.
HELP <command>	Return information on specified command.
LIST <name>	List information if name is a file or list files if name is a directory.
MODE <mode>	Transfer mode (S=stream, B=block, C=compressed).
MKD <directory>	Create specified directory on server.
NLST <directory>	List contents of specified directory.
NOOP	Cause no action other than acknowledgement from server.
PASS <password>	Password for system log-in.
PASV	Request server wait for data connection.
PORT <address>	IP address and two-byte system port ID.
PWD	Display current working directory.
QUIT	Log off from the FTP server.
REIN	Reinitialize connection to log-in status.

Command	*Description*
REST <offset>	Restart file transfer from given offset.
RETR <filename>	Retrieve (copy) file from server.
RMD <directory>	Remove specified directory on server.
RNFR <old path>	Rename from old path.
RNTO <new path>	Rename to new path.
SITE <params>	Site specific parameters provided by server.
SMNT <pathname>	Mount the specified file structure.
STAT <directory>	Return information on current process or directory.
STOR <filename>	Store (copy) file to server.
STOU <filename>	Store file to server name.
STRU <type>	Data structure (F=file, R=record, P=page).
SYST	Return operating system used by server.
TYPE <data type>	Data type (A=ASCII, E=EBCDIC, I=binary).
USER <username>	User name for system log-in.

Messages

FTP messages are responses to FTP commands and consist of a response code followed by explanatory text. Standard FTP messages are as follows:

Response Code	*Explanatory Text*
110	Restart marker at MARK yyyy=mmmm (new file pointers).
120	Service ready in nnn minutes.
125	Data connection open, transfer starting.
150	Open connection.
200	OK.
202	Command not implemented.
211	(System status reply).
212	(Directory status reply).
213	(File status reply).
214	(Help message reply).
215	(System type reply).
220	Service ready.
221	Log off network.
225	Data connection open.
226	Close data connection.
227	Enter passive mode (IP address, port ID).
230	Log on network.

Response Code	*Explanatory Text*
250	File action completed.
257	Path name created.
331	Password required.
332	Account name required.
350	File action pending.
421	Service shutting down.
425	Cannot open data connection.
426	Connection closed.
450	File unavailable.
451	Local error encountered.
452	Insufficient disk space.
500	Invalid command.
501	Bad parameter.
502	Command not implemented.
503	Bad command sequence.
504	Parameter invalid for command.
530	Not logged onto network.
532	Need account for storing files.
550	File unavailable.
551	Page type unknown.
552	Storage allocation exceeded.
553	File name not allowed.

TFTP

IETF RFC 1350 1992-07 http://www.cis.ohio-state.edu/htbin/rfc/rfc1350.html
IETF RFC 783 http://www.cis.ohio-state.edu/htbin/rfc/rfc783.html

The Trivial File Transfer Protocol (TFTP) uses UDP. TFTP supports file writing and reading; it does not support directory service of user authorization.

Commands

The following are TFTP commands:

Command	*Description*
Read Request	Request to read a file.
Write Request	Request to write to a file.
File Data	Transfer of file data.
Data Acknowledge	Acknowledgement of file data.
Error	Error indication.

Parameters

TFTP Read and Write Request commands use the following parameters:

Parameter	*Description*	
Filename	The name of the file, expressed in quotes, where the protocol is to perform the read or write operation.	
Mode	Datamode. The format of the file data that the protocol is to transfer. The following formats are possible:	
	NetASCII	Standard ASCII character format.
	Octet	Eight-bit binary data.
	Mail	Standard ASCII character format with username in place of filename.

TFTP data and data acknowledge commands use the following parameters:

Command	*Description*
Block	Block number or sequence number of the current frame of file data.
Data	First part of the file data displayed for TFTP data frames.

Command	*Description*
TFTP Errors	TFTP error frames contain an error code in parentheses followed by the error message, as follows: (0000) Unknown Error. (0001) File not found. (0002) Access violation. (0003) Out of disk space. (0004) Illegal TFTP operation. (0005) Unknown Transfer ID. (0006) Filename already exists. (0007) Unknown user.

Finger

RFC 1288 http://www.cis.ohio-state.edu/htbin/rfc/rfc1288.html

The Finger user information protocol is a simple protocol which provides an interface to a remote user information program. It is a protocol for the exchange of user information, based on the Transmission Control Protocol, using TCP port 79 decimal (117 octal). The local host opens a TCP connection to a remote host on the Finger port. An RUIP becomes available on the remote end of the connection to process the request. The local host sends the RUIP a one line query based upon the Finger query specification, and waits for the RUIP to respond. The RUIP receives and processes the query, returns an answer, then initiates the close of the connection. The local host receives the answer and the close signal, then proceeds closing its end of the connection.

The Finger protocol displays data. Any data transferred must be in ASCII format, with no parity, and with lines ending in CRLF (ASCII 13 followed by ASCII 10). This excludes other character formats such as EBCDIC, etc. This also means that any characters between ASCII 128 and ASCII 255 should truly be international data, not 7-bit ASCII with the parity bit set. Note: if ASCII 13 followed by ASCII 10 transferred, the character won't display (because the only meaning is to end the line).

Gopher

RFC 1436 http://www.cis.ohio-state.edu/htbin/rfc/rfc1436.html

The Internet Gopher protocol and software follow a client-server model. This protocol assumes a reliable data stream; TCP is assumed. Gopher servers listen on port 70 (port 70 is assigned to Internet Gopher by IANA). Documents reside on many autonomous servers on the Internet. Users run client software on their desktop systems, connecting to a server and sending the server a selector (a line of text, which may be empty) via a TCP connection at a well-known port. The server responds with a block of text terminated by a period on a line by itself and closes the connection. No state is retained by the server.

The first character on each line tells whether the line describes a document, directory, or search service (characters '0', '1', '7'; there are a handful more of these characters described later). The succeeding characters up to the tab form a user display string to be shown to the user for use in selecting this document (or directory) for retrieval. The first character of the line is really defining the type of item described on this line. In nearly every case, the Gopher client software will give the users some sort of idea about what type of item this is (by displaying an icon, a short text tag, or the like).

The characters following the tab, up to the next tab form a selector string that the client software must send to the server to retrieve the document (or directory listing). The selector string should mean nothing to the client software; it should never be modified by the client. In practice, the selector string is often a pathname or other file selector used by the server to locate the item desired. The next two tab delimited fields denote the domain-name of the host that has this document (or directory), and the port at which to connect. If there are yet other tab delimited fields, the basic Gopher client should ignore them. A CR LF denotes the end of the item.

Item type characters

The client software decides what items are available by looking at the first character of each line in a directory listing. Augmenting this list can extend the protocol. A list of defined item-type characters follows:

0 Item is a file.
1 Item is a directory.
2 Item is a CSO phone-book server.
3 Error.

4 Item is a BinHexed Macintosh file.
5 Item is DOS binary archive of some sort. The client must read until the TCP connection closes.
6 Item is a UNIX uuencoded file.
7 Item is an Index-Search server.
8 Item points to a text-based telnet session.
9 Item is a binary file. The client must read until the TCP connection closes.
+ Item is a redundant server
T Item points to a text-based tn3270 session.
g Item is a GIF format graphics file.
I Item is some kind of image file. The client decides how to display.

Characters '0' through 'Z' are reserved. Local experiments should use other characters. Machine-specific extensions are not encouraged. Note that for type 5 or type 9 the client must be prepared to read until the connection closes. There will be no period at the end of the file; the contents of these files are binary and the client must decide what to do with them based perhaps on the .xxx extension.

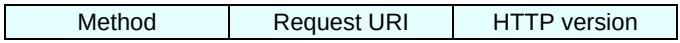
HTTP

RFC 1945 http://www.cis.ohio-state.edu/htbin/rfc/rfc1945.html

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. Messages are passed in a format similar to that used by Internet Mail and the Multipurpose Internet Mail Extensions (MIME).

Request Packet

The format of the Request packet header is shown in the following illustration:



HTTP request packet structure

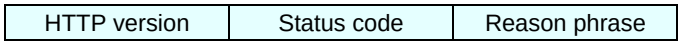
Method
The method to be performed on the resource.

Request-URI
The Uniform Resource Identifier, the resource upon which to apply the request, i.e. the network resource.

HTTP version
The HTTP version being used.

Response Packet

The format of the Response packet header is shown in the following illustration:



HTTP response packet structure

HTTP version
The HTTP version being used.

Status-code
A 3 digit integer result code of the attempt to understand and satisfy the request.

Reason-phrase
A textual description of the status code.

S-HTTP

draft-ietf-wts-shttp-06

Secure HTTP (S-HTTP) provides secure communication mechanisms between an HTTP client-server pair in order to enable spontaneous commercial transactions for a wide range of applications. S-HTTP provides a flexible protocol that supports multiple orthogonal operation modes, key management mechanisms, trust models, cryptographic algorithms and encapsulation formats through option negotiation between parties for each transaction. Syntactically, S-HTTP messages are the same as HTTP, consisting of a request or status line followed by headers and a body. However, the range of headers is different and the bodies are typically cryptographically enhanced.

IMAP4

RFC 2060 http://www.cis.ohio-state.edu/htbin/rfc/rfc2060.html

The Internet Message Access Protocol, Version 4 revision 1 (IMAP4) allows a client to access and manipulate electronic mail messages on a server. IMAP4 permits manipulation of remote message folders, called mailboxes, in a way that is functionally equivalent to local mailboxes. IMAP4 also provides the capability for an offline client to resynchronize with the server.

IMAP4 includes operations for creating, deleting, and renaming mailboxes; checking for new messages; permanently removing messages; setting and clearing flags; parsing; searching; and selective fetching of message attributes, texts, and portions thereof. Messages in IMAP4 are accessed by the use of numbers. These numbers are either message sequence numbers or unique identifiers.

IMAP4 consists of a sequence of textual messages which contain commands, status messages, etc. Each message ends with <crlf>(carriage return and line feed). For example:

```
Server Message: "a002 OK [READ-WRITE] SELECT
completed<crlf>"
```

```
Client Message: "a001 login mrc secret<crlf>"
```

There are no other predefined fields.

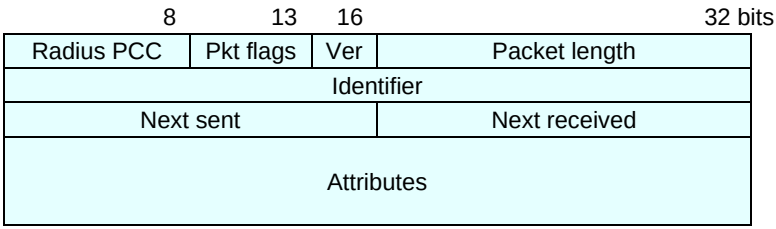
IPDC

Internet Drafts: – draft-taylor-ipdc-00.txt and draft-calhoun-diameter-07.txt.
http://www.ietf.org/internet-drafts/draft-taylor-ipdc-00.txt
http://www.ietf.org/internet-drafts/draft-calhoun-diameter-07.txt

The IP Device Control (IPDC) is a family of protocols which is proposed as a protocol suite, components of which can be used individually or together to perform connection control, media control, and signalling transports. It fulfils a need for one or more protocols to control gateway devices which sit at the boundary between the circuit- switched telephone network and the internet and terminate circuit- switched trunks. Examples of such devices include network access servers and voice-over-IP gateways. The need for a control protocol separate from call signalling, arises when the service control logic needed to process calls lies partly or wholly outside the gateway devices.

IPDC was built on the base structure provided by the DIAMETER protocol which was specifically written for authentication, authorization and accounting applications.

There are two different types of IPDC/DIAMETER messages: header-only messages and messages containing Attribute-Value Pairs (AVPs) in addition to headers. Header-only messages are used for explicitly acknowledging packets to the peer. An AVP is a data object encapsulated in a header. The general format of the header is shown in the following illustration:



IPDC header structure

Radius PCC

Radius packet compatibility code, used for Radius backward compatibility. In order to easily distinguish DIAMETER/IPDC messages from Radius, a special value has been reserved and allows an implementation to support

both protocols concurrently using the first octet in the header. The Radius PCC field must be set to 254 for DIAMETER/IPDC messages.

Pkt flags

Packet flags. Used to identify any options. This field must be initialized to zero. The Window-Present flag may be set (0x1), thus indicating that the Next Send and Next Received fields are present. This flag must be set unless the underlying layer provides reliability (i.e., TCP).

Version

Indicates the version number associated with the packet received. This field is set to 1 to indicate IPDC version 1.

Packet length

Indicates the length of the message including the header fields. Thus the message AVP content cannot exceed 65,528 octets. For messages received via UDP, octets outside the range of the length field should be treated as padding and are ignored upon receipt.

Identifier

Aids in matching requests and replies.

Next sent (Ns)

Present when the Window-Present bit is set in the header flags. The Next Send (Ns) is copied from the send sequence number state variable, Ss, at the time the message is transmitted.

Next received

This field is present when the Window-Present bit is set in the header flags. Nr is copied from the receive sequence number state variable, Sr, and indicates the sequence number, Ns, +1 of the highest (modulo 2^16) in-sequence message received.

Attributes

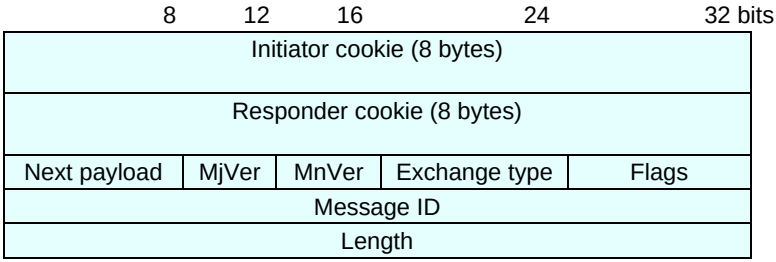
IPDC Attributes carry the specific commands and parameters which must be exchanged between IPDC protocol endpoints to perform the tasks associated with Media Gateway control.

ISAKMP

RFC2408 http://www.cis.ohio-state.edu/htbin/rfc/rfc2408.html

The Internet Security Association and Key Management Protocol, version 4rev1 (ISAKMP), defines procedures and packet formats to establish, negotiate, modify and delete Security Associations (SA). SAs contain all the information required for execution of various network security services, such as the IP layer services (such as header authentication and payload encapsulation), transport or application layer services, or self-protection of negotiation traffic. ISAKMP defines payloads for exchanging key generation and authentication data. These formats provide a consistent framework for transferring key and authentication data which is independent of the key generation technique, encryption algorithm and authentication mechanism.

The format of the header is shown in the following illustration:



ISAKMP header structure

Initiator cookie

Cookie of entity that initiated SA establishment, SA notification, or SA deletion.

Responder cookie

Cookie of entity that is responding to an SA establishment, SA notification, or SA deletion.

Next payload

Indicates the type of the first payload in the message. Possible types are:

0 None.
1 Security Association (SA).
2 Proposal (P).

3 Transform (T).
4 Key Exchange (KE).
5 Identification (ID).
6 Certificate (CERT).
7 Certificate Request (CR).
8 Hash (HASH).
9 Signature (SIG).
10 Nonce (NONCE).
11 Notification (N).
12 Delete (D).
13 Vendor ID (VID).
14 - 127 Reserved.
128 - 255 Private use.

MjVer

Major Version, indicates the major version of the ISAKMP protocol in use. Implementations based on RFC2408 must set the Major Version to 1. Implementations based on previous versions of ISAKMP Internet- Drafts must set the Major Version to 0. Implementations should never accept packets with a major version number larger than its own.

MnVer

Minor Version - indicates the minor version of the ISAKMP protocol in use. Implementations based on RFC2408 must set the minor version to 0. Implementations based on previous versions of ISAKMP Internet- Drafts must set the minor version to 1. Implementations should never accept packets with a minor version number larger than its own.

Exchange Type

The type of exchange being used. This dictates the message and payload orderings in the ISAKMP exchanges. Possible values are:

0 None
1 Base
2 Identity Protection
3 Authentication Only
4 Aggressive
5 Informational
6 - 31 ISAKMP Future Use
32 - 239 DOI Specific Use
240 - 255 Private Use

Flags

Specific options that are set for the ISAKMP exchange.
E(ncryption bit) (bit 0) - Specifies that all payloads following the header are encrypted using the encryption algorithm identified in the ISAKMP SA.
C(ommit bit) (bit 1) - Signals key exchange synchronization. It is used to ensure that encrypted material is not received prior to completion of the SA establishment.
A(uthentication Only Bit) (bit 2) - Intended for use with the Informational Exchange with a Notify payload and will allow the transmission of information with integrity checking, but no encryption.
All remaining bits are set to 0 before transmission.

Message ID

Unique Message Identifier used to identify protocol state during Phase 2 negotiations. This value is randomly generated by the initiator of the Phase 2 negotiation. In the event of simultaneous SA establishments (i.e., collisions), the value of this field will likely be different because they are independently generated and, thus, two security associations will progress toward establishment. However, it is unlikely there will be absolute simultaneous establishments. During Phase 1 negotiations, the value must be set to 0.

Length

Length of total message (header + payloads) in octets. Encryption can expand the size of an ISAKMP message.

NTP

RFC 1305 http://www.cis.ohio-state.edu/htbin/rfc/rfc1305.html

The Network Time Protocol (NTP) is a time synchronization system for computer clocks through the Internet network. It provides the mechanisms to synchronize time and coordinate time distribution in a large, diverse internet operating at rates from mundane to light wave. It uses a returnable time design in which a distributed sub network of time servers, operating in a self-organizing, hierarchical master-slave configuration, synchronize logical clocks within the sub network and to national time standards via wire or radio.

The format of the header is shown in the following illustration:

LI	VN	Mode	Stratum	Poll	Precision	
2	3	3	7	6	7	bits

NTP header structure

LI Leap Indicator

A 2-bit code warning of impending leap-second to be inserted at the end of the last day of the current month. Bits are coded as follows:

00 No warning.
01 +1 second (following minute has 61 seconds).
10 -1 second (following minute has 59 seconds).
11 Alarm condition (clock not synchronized).

VN

Version number 3 bit code indicating the version number.

Mode

The mode: This field can contain the following values:

0 Reserved.
1 Symmetric active.
2 Symmetric passive.
3 Client.
4 Server.
5 Broadcast.
6 NTP control message.

Stratum

An integer identifying the stratum level of the local clock. Values are defined as follows:

0 Unspecified.
1 Primary reference (e.g. radio clock).
2...n Secondary reference (via NTP).

Poll

Signed integer indicating the maximum interval between successive messages, in seconds to the nearest power of 2.

Precision

Signed integer indicating the precision of the local clock, in seconds to the nearest power of 2.

POP3

RFC 1939 http://www.cis.ohio-state.edu/htbin/rfc/rfc1939.html

The Post Office Protocol version 3 (POP3) is intended to permit a workstation to dynamically access a maildrop on a server host. It is usually used to allow a workstation to retrieve mail that the server is holding for it.

POP3 transmissions appear as data messages between stations. The messages are either command or reply messages.

Radius

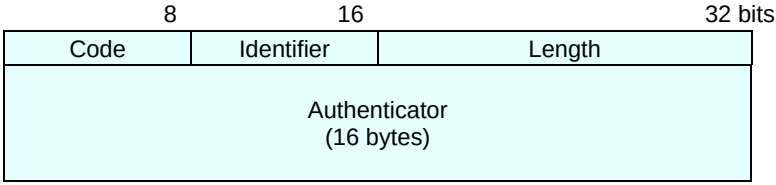
RFC 2138 http://www.cis.ohio-state.edu/htbin/rfc/rfc2138.html
RFC 2139 http://www.cis.ohio-state.edu/htbin/rfc/rfc2139.html

Radius is a protocol which manages dispersed serial line and modem pools for large numbers of users. Since modem pools are by definition a link to the outside world, they require careful attention to security, authorization and accounting. This is achieved by managing a single database of users, which allows for authentication (verifying user name and password) as well as configuration information detailing the type of service to deliver to the user (for example, SLIP, PPP, telnet, rlogin).

Key features of RADIUS include:

- Client/server model.
- Network security.
- Flexible authentication mechanisms.
- Extensible protocol.

The format of the header is shown in the following illustration:



Radius header structure

Code
The message type.

Identifier
The identifier matches requests and replies.

Length
The message length including the header.

Authenticator
A field used to authenticate the reply from the radius server and in the password hiding algorithm.

RLOGIN

Remote LOGIN (RLOGIN) allows UNIX users of one machine to connect to other UNIX systems across an Internet and interact as if their terminals are directly connected to the machines. This protocol offers essentially the same services as TELNET.

RTSP

RFC 2326 http://www.cis.ohio-state.edu/htbin/rfc/rfc2326.html

RTSP (Real Time Streaming Protocol) is an application level protocol for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live data feeds and stored clips. This protocol is intended to control multiple data delivery sessions, provide a means for choosing delivery channels such as UDP, multicast UDP and TCP, and provide a means for choosing delivery mechanisms based upon RTP.

The streams controlled by RTSP may use RTP, but the operation of RTSP does not depend on the transport mechanism used to carry continuous media. The protocol is intentionally similar in syntax and operation to HTTP/1.1 so that extension mechanisms to HTTP can in most cases also be added to RTSP. However, RTSP differs in a number of important aspects from HTTP:

- RTSP introduces a number of new methods and has a different protocol identifier.
- An RTSP server needs to maintain state by default in almost all cases, as opposed to the stateless nature of HTTP.
- Both an RTSP server and client can issue requests.
- Data is carried out-of-band by a different protocol.
- RTSP is defined to use ISO 10646 (UTF-8) rather than ISO 8859-1, consistent with current HTML internationalization efforts.
- The Request-URI always contains the absolute URI. Because of backward compatibility with an historical blunder, HTTP/1.1 carries only the absolute path in the request and puts the host name in a separate header field.

This makes virtual hosting easier, where a single host with one IP address hosts several document trees.

SMTP

RFC 821 http://www.cis.ohio-state.edu/htbin/rfc/rfc821.html

The Simple Mail Transfer Protocol (SMTP) is a mail service modeled on the FTP file transfer service. SMTP transfers mail messages between systems and provides notification regarding incoming mail.

Commands

SMTP commands are ASCII messages sent between SMTP hosts. Possible commands are as follows:

Command	*Description*
DATA	Begins message composition.
EXPN <string>	Returns names on the specified mail list.
HELO <domain>	Returns identity of mail server.
HELP <command>	Returns information on the specified command.
MAIL FROM <host>	Initiates a mail session from host.
NOOP	Causes no action, except acknowledgement from server.
QUIT	Terminates the mail session.
RCPT TO <user>	Designates who receives mail.
RSET	Resets mail connection.
SAML FROM <host>	Sends mail to user terminal and mailbox.
SEND FROM <host>	Sends mail to user terminal.
SOML FROM <host>	Sends mail to user terminal or mailbox.
TURN	Switches role of receiver and sender.
VRFY <user>	Verifies the identity of a user.

Messages

SMTP response messages consist of a response code followed by explanatory text, as follows:

Response Code	*Explanatory Text*
211	(Response to system status or help request).
214	(Response to help request).
220	Mail service ready.
221	Mail service closing connection.
250	Mail transfer completed.
251	User not local, forward to <path>.
354	Start mail message, end with <CRLF><CRLF>.
421	Mail service unavailable.
450	Mailbox unavailable.
451	Local error in processing command.
452	Insufficient system storage.
500	Unknown command.
501	Bad parameter.
502	Command not implemented.
503	Bad command sequence.
504	Parameter not implemented.
550	Mailbox not found.
551	User not local, try <path>.
552	Storage allocation exceeded.
553	Mailbox name not allowed.
554	Mail transaction failed.

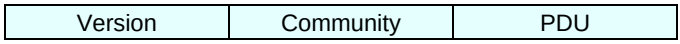
SNMP

RFC 1157: http://www.cis.ohio-state.edu/htbin/rfc/rfc1157.html

The Internet community developed the Simple Network Management Protocol (SNMP) to allow diverse network objects to participate in a global network management architecture. Network managing systems can poll network entities implementing SNMP for information relevant to a particular network management implementation. Network management systems learn of problems by receiving traps or change notices from network devices implementing SNMP.

SNMP Message Format

SNMP is a session protocol which is encapsulated in UDP. The SNMP message format is shown below:



SNMP message format

Version

SNMP version number. Both the manager and agent must use the same version of SNMP. Messages containing different version numbers are discarded without further processing.

Community

Community name used for authenticating the manager before allowing access to the agent.

PDU

There are five different PDU types: GetRequest, GetNextRequest, GetResponse, SetRequest, and Trap. A general description of each of these is given in the next section.

PDU Format

The format for GetRequest, GetNext Request, GetResponse and SetRequest PDUs is shown here.

PDU type	Request ID	Error status	Error index	Object 1, value 1	Object 2, value 2	…

SNMP PDU format

PDU type

Specifies the type of PDU:

0 GetRequest.
1 GetNextRequest.
2 GetResponse.
3 SetRequest.

Request ID

Integer field which correlates the manager's request to the agent's response.

Error status

Enumerated integer type that indicates normal operation or one of five error conditions. The possible values are:

0 noError: Proper manager/agent operation.
1 tooBig: Size of the required GetResponse PDU exceeds a local limitation.
2 noSuchName: The requested object name does not match the names available in the relevant MIB View.
3 badValue: A SetRequest contains an inconsistent type, length and value for the variable.
4 readOnly: Not defined in RFC1157.
5 genErr: Other errors, which are not explicitly defined, have occurred.

Error index

Identifies the entry within the variable bindings list that caused the error.

Object/value

Variable binding pair of a variable name with its value.

Trap PDU Format

The format of the Trap PDU is shown below:

PDU type	Enterp	Agent addr	Gen trap	Spec trap	Time stamp	Obj 1, Val 1	Obj 1, Val 1	…

SNMP trap PDU

PDU type
Specifies the type of PDU (4=Trap).

Enterprise
Identifies the management enterprise under whose registration authority the trap was defined.

Agent address
IP address of the agent, used for further identification.

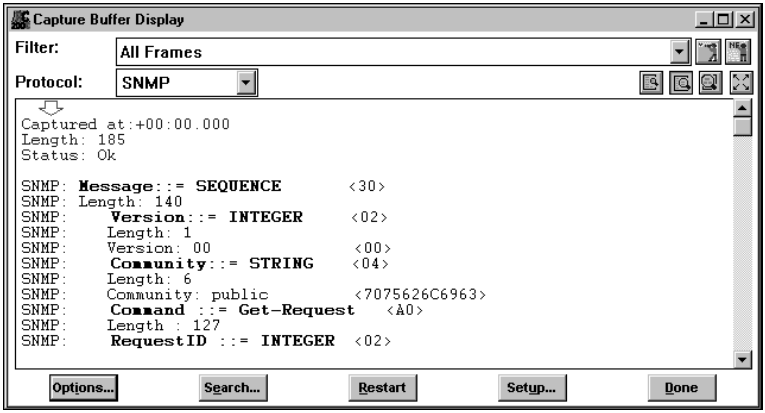
Generic trap type
Field describing the event being reported. The following seven values are defined:

0 coldStart: Sending protocol entity has reinitialized, indicating that the agent's configuration or entity implementation may be altered.
1 warmStart: Sending protocol has reinitialized, but neither the agent's configuration nor the protocol entity implementation has been altered.
2 linkDown: A communication link has failed.
3 linkUp: A communication link has come up.
4 authenticationFailure: The agent has received an improperly authenticated SNMP message from the manager, i.e., community name was incorrect.
5 egpNeighborLoss: An EGP peer neighbor is down.
6 enterpriseSpecific: A non-generic trap has occurred which is further identified by the Specific Trap Type and Enterprise fields.

Specific trap type
Used to identify a non-generic trap when the Generic Trap Type is enterpriseSpecific.

Timestamp
Value of the sysUpTime object, representing the amount of time elapsed between the last (re-)initialization and the generation of that Trap.

Object/value
Variable binding pair of a variable name with its value.


Capture Buffer Display
Filter:
All Frames
Protocol:
SNMP
Captured at:+00:00.000
Length: 185
Status: Ok
SNMP: Message::= SEQUENCE <30>
SNMP: Length: 140
SNMP: Version::= INTEGER <02>
SNMP: Length: 1
SNMP: Version: 00 <00>
SNMP: Community::= STRING <04>
SNMP: Length: 6
SNMP: Community: public <7075626C6963>
SNMP: Command ::= Get-Request <A0>
SNMP: Length : 127
SNMP: RequestID ::= INTEGER <02>
Options...
Search...
Restart
Setup...
Done

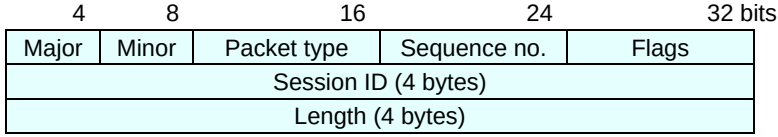
SNMP decode

TACACS+

draft-grant-tacacs-02.txt
http://www.ietf.org/internet-drafts/draft-grant-tacacs-02.txt
RFC 1492 http://www.cis.ohio-state.edu/htbin/rfc/rfc1492.html

TACACS+ (Terminal Access Controller Access Control System) is a protocol providing access control for routers, network access servers and other networked computing devices via one or more centralized servers. TACACS+ provides separate authentication, authorization and accounting services.

The format of the header is shown in the following illustration:



TACACS+ header structure

Major version
The major TACACS+ version number.

Minor version
The minor TACACS+ version number. This is intended to allow revisions to the TACACS+ protocol while maintaining backwards compatibility.

Packet type
Possible values are:
TAC_PLUS_AUTHEN:= 0x01 (Authentication).
TAC_PLUS_AUTHOR:= 0x02 (Authorization).
TAC_PLUS_ACCT:= 0x03 (Accounting).

Sequence number
The sequence number of the current packet for the current session. The first TACACS+ packet in a session must have the sequence number 1 and each subsequent packet will increment the sequence number by one. Thus clients only send packets containing odd sequence numbers, and TACACS+ daemons only send packets containing even sequence numbers.

Flags
This field contains various flags in the form of bitmaps. The flag values signify whether the packet is encrypted.

Session ID
The ID for this TACACS+ session.

Length
The total length of the TACACS+ packet body (not including the header).

TELNET

IETF RFC 854 1983-05 http://www.cis.ohio-state.edu/htbin/rfc/rfc854.html
IETF RFC 855 1983-05 http://www.cis.ohio-state.edu/htbin/rfc/rfc855.html
IETF RFC 857 1983-05 http://www.cis.ohio-state.edu/htbin/rfc/rfc857.html

TELNET is the terminal emulation protocol of TCP/IP. Modern TELNET is a versatile terminal emulation due to the many options that have evolved over the past twenty years. Options give TELNET the ability to transfer binary data, support byte macros, emulate graphics terminals, and convey information to support centralized terminal management.

TELNET uses the TCP transport protocol to achieve a virtual connection between server and client. After connecting, TELNET server and client enter a phase of option negotiation that determines the options that each side can support for the connection. Each connected system can negotiate new options or renegotiate old options at any time. In general, each end of the TELNET connection attempts to implement all options that maximize performance for the systems involved.

In a typical implementation, the TELNET client sends single keystrokes, while the TELNET server can send one or more lines of characters in response. Where the Echo option is in use, the TELNET server echoes all keystrokes back to the TELNET client.

Dynamic Mode Negotiation

During the connection, enhanced characteristics other than those offered by the NVT may be negotiated either by the user or the application. This task is accomplished by embedded commands in the data stream. TELNET command codes are one or more octets in length and are preceded by an interpret as command (IAC) character, which is an octet with each bit set equal to one (FF hex). The following are the TELNET command codes:

Commands	*Code No.*		*Description*
	Dec	*Hex*	
data			All terminal input/output data.
End subNeg	240	FO	End of option subnegotiation command.
No Operation	241	F1	No operation command.
Data Mark	242	F2	End of urgent data stream.

Commands	*Code No.*		*Description*
	Dec	*Hex*	
Break	243	F3	Operator pressed the Break key or the Attention key.
Int process	244	F4	Interrupt current process.
Abort output	245	F5	Cancel output from current process.
You there?	246	F6	Request acknowledgment.
Erase char	247	F7	Request that operator erase the previous character.
Erase line	248	F8	Request that operator erase the previous line.
Go ahead!	249	F9	End of input for half-duplex connections.
SubNegotiate	250	FA	Begin option subnegotiation.
Will Use	251	FB	Agreement to use the specified option.
Won't Use	252	FC	Reject the proposed option.
Start use	253	FD	Request to start using specified option.
Stop Use	254	FE	Demand to stop using specified option.
IAC	255	FF	Interpret as command.

Each negotiable option has an ID, which immediately follows the command for option negotiation, that is, IAC, command, option code. The following is a list of TELNET option codes:

Option ID		*Option Codes*	*Description*
Dec	*Hex*		
0	0	Binary Xmit	Allows transmission of binary data.
1	1	Echo Data	Causes server to echo back all keystrokes.
2	2	Reconnect	Reconnects to another TELNET host.
3	3	Suppress GA	Disables Go Ahead! command.
4	4	Message Sz	Conveys approximate message size.
5	5	Opt Status	Lists status of options.
6	6	Timing Mark	Marks a data stream position for reference.
7	7	R/C XmtEcho	Allows remote control of terminal printers.
8	8	Line Width	Sets output line width.
9	9	Page Length	Sets page length in lines.
10	A	CR Use	Determines handling of carriage returns.
11	B	Horiz Tabs	Sets horizontal tabs.
12	C	Hor Tab Use	Determines handling of horizontal tabs.

Option ID		*Option Codes*	*Description*
Dec	*Hex*		
13	D	FF Use	Determines handling of form feeds.
14	E	Vert Tabs	Sets vertical tabs.
15	F	Ver Tab Use	Determines handling of vertical tabs.
16	10	Lf Use	Determines handling of line feeds.
17	11	Ext ASCII	Defines extended ASCII characters.
18	12	Logout	Allows for forced log-off.
19	13	Byte Macro	Defines byte macros.
20	14	Data Term	Allows subcommands for Data Entry to be sent.
21	15	SUPDUP	Allows use of SUPDUP display protocol.
22	16	SUPDUP Outp	Allows sending of SUPDUP output.
23	17	Send Locate	Allows terminal location to be sent.
24	18	Term Type	Allows exchange of terminal type information.
25	19	End Record	Allows use of the End of record code (0xEF).
26	1A	TACACS ID	User ID exchange used to avoid more than 1 log-in.
27	1B	Output Mark	Allows banner markings to be sent on output.
28	1C	Term Loc#	A numeric ID used to identify terminals.
29	1D	3270 Regime	Allows emulation of 3270 family terminals.
30	1E	X.3 PAD	Allows use of X.3 protocol emulation.
31	1F	Window Size	Conveys window size for emulation screen.
32	20	Term Speed	Conveys baud rate information.
33	21	Remote Flow	Provides flow control (XON, XOFF).
34	22	Linemode	Provides linemode bulk character transactions.
255	FF	Extended options list	Extended options list.

X-Window

The X-Window protocol provides a remote windowing interface to distributed network applications. It is an application layer protocol which uses TCP/IP or DECnet protocols for transport.

The X-Window networking protocol is client-server based, where the server is the control program running on the user workstation and the client is an application running elsewhere on the network. An X-server control program running on a workstation can simultaneously handle display windows for multiple applications, with each application asynchronously updating its window with information carried by the X-Window networking protocol.

To provide user interaction with remote applications, the X-server program running on the workstation generates events in response to user input such as mouse movement or a keystroke. When multiple applications display, the system sends mouse movements or click events to the application currently highlighted by the mouse pointer. The current input focus selects which application receives keystroke events. In certain cases, applications can also generate events directed at the X-server control program.

Request and Reply Frames

Request and reply frames can use the following commands:

Command	*Description*
BackRGB	Background colors listed in red, green and blue components.
BackPM	Pixel map used for the window background.
BellPitch	Bell pitch.
BellVol	Bell volume in percent.
BM	Bit mask assigned to a drawable item.
BordPM	Border pixel map. Pixel map used for the window border.
b	Border width of the drawable item.
Click	Key click volume in percent.
Ord	Click order. Drawable clip order, as <Unsorted>, <Y-sorted>, <YX-sorted> or <YX-banded>.
CMap	Color map. Code representing the colors in use for a drawable.

Command	*Description*
CID	Context ID. Identifier for a particular graphics context.
Cur	Cursor. Reference code identifying a specific cursor.
d	Depth. Current window depth.
DD	Destination drawable. Target item in a bitmap copy.
D	Drawable. Reference code used to identify a specific window or pixel map.
Exp	Exposures. Drawable currently exposed.
Fam	Protocol family in use, as Internet, DECnet, or CHAOSnet.
Font	Reference code used to specify a font.
Font(a,d)	Font ascent/descent. The vertical bounds of a font.
ForeRGB	Foreground colors listed in red, green, and blue components.
Fmt	Format of the current window.
GC	Graphics context. Reference code used to identify a particular graphical definition.
h	Height of the drawable item.
Key	Key code. Specific key code value.
KeySym	Code used to identify the family of key codes in use.
MinOp	X-Windows minor operation code.
MajOp	X-Windows major operation code.
N	Number of drawable items in the list.
P	Parent window. Window that produced the current window.
PixMap	Pixel map. Reference code used to identify a bitmap region.
p	Plane. Bit plane in use.
PM	Plane max. Bit plane mask assigned to a drawable item.
Prop	Property. Specified window property.
SW	Sibling window. Window produced from this window.
SD	Source drawable. Source item in a bitmap copy.
T/O	Screen saver time out.
Typ	Type of current window.

Command	*Description*
w	Width of drawable item.
W	Window. Reference code used to identify a particular window.
X	X-coordinate for a drawable item.
Y	Y-coordinate for a drawable item.

Event Frames

Event frames can have the following commands:

Command	*Description*
Btn	Button number pressed.
C	Child window associated with the event.
F	Event flags. Set flags display in upper-case and inactive flags display in lower-case: f,F Input focus applies to the event. s,S Event is on the same screen.
E(x,y)	Event location. The X and Y coordinates of the event.
E	Event window. Window where the event occurred.
Key	Key number. Number associated with the pressed key.
O	Owner of the window associated with the event.
R	Root window associated with the event.
R(x,y)	Root location. X and Y coordinates of the root position.
SN	Sequence number used to serialize events.